4.1 A Third-Generation 65nm 16-Core 32-Thread Plus 32-Scout-Thread CMT SPARC® Processor

Marc Tremblay, Shailender Chaudhry

Sun Microsystems, Santa Clara, CA

The goals for this high-end commercial microprocessor are high throughput and high single-thread performance, mainframe-class reliability, hardware transactional memory, and linear scalability. We show how these goals are met by the logical and physical design of this 2.3GHz 396mm² 16-core 32-thread plus 32-scout-thread microprocessor.

High throughput is achieved by maximizing the number of multi-threaded cores that are accommodated with appropriate cache structures on a single 65nm die. In total, 16 small (14mm²) and relatively low-power cores (~10W), supporting a new form of multi-threading, deliver 32 threads. Traditional out-of-order execution is discarded due to the large CAMs needed to implement a large instruction window (e.g., 512 rows of over 100b plus recovery structures) and the large re-order buffer necessary to hide current memory latency (500 to 1000 cycles). Another discarded option is a flat topology with redundant instruction cache (I$) copies and a large crossbar or a multi-stage network connecting cores. The system implemented is a hierarchical structure, where a cluster of 4 cores shares an I$, 2 data caches (D$), two floating-point units (FPU), and a port to a dual 5×5 switch, achieving a high temporal utilization of components (Figure 4.1.1). The 32KB + 8KB (predecoded bits), 4-way pseudo-LRU-shared I$ delivers up to 16 instructions per cycle to a single core, satisfying instruction bandwidth requirements. Given the typically high re-utilization of code across processes and threads on a server, this consolidation yields a significant savings in area and power (e.g., 4 I$ instead of 16). Similarly, a dual-banked 32KB 4-way pseudorandom-replacement D$ is shared amongst 2 cores, resulting in significant savings (8 D$ instead of 16), yet still sustains one load per cycle per core.

Maximizing the utilization of power-hungry FPUs leads to a fully pipelined fully interleaved FPU shared across 2 cores, saving 8 FPUs while sacrificing little application throughput. The dual 5×5 switches provide five 298b read ports and five 140b write ports at full processor frequency. In comparison, a 17×17 crossbar is at least 8× larger and would only sustain a fraction of its peak bandwidth under normal workloads. A 2MB 4-bank 8-way pseudo-LRU L2$ with hashed physical address (per thread) supports 4 independent requests simultaneously. L2$ misses are forwarded to non-blocking memory interface units, which collectively support 128 outstanding requests to external L3$/memory controllers.

The floorplan of this 396mm² die in 11M 65nm CMOS is suggested by the sharing of structures and the core hierarchy of the architecture. The core cluster floorplan is shown in Fig. 4.1.2 and Fig. 4.1.3 shows a full-chip microphotograph. The crossbar and L2$, shared by all cores, are in the middle of the die. The details on the implementation are provided in [1].

To ensure high instruction-level parallelism and high memory-level parallelism, we created a new power-efficient pipeline capable of deep speculation (Fig. 4.1.4.). As opposed to traditional out-of-order execution, where all instructions, including completed ones, are kept in power-hungry CAMs, only non-completed instructions are kept in a 128-deep SRAM structure. This effectively compresses a classic window of up to 1000 instructions into a small low-power structure, allowing 16 instances of the pipeline on a single chip. The pipeline is capable of using idle structures normally used by other threads, further enhancing single thread performance.

The processor uses a checkpoint-based architecture. Upon encountering a long-latency instruction such as a load miss, the pipeline takes a checkpoint, proceeding as if it had completed the instruction and continuing program execution [2]. The load destination is marked as not available (NA) and is propagated through dependence chains. Clean register re-definitions later make NA registers available again. When data for a long-latency instruction becomes available, the pipeline proceeds from the checkpoint. This allows the pipe to issue independent load misses under the first long-latency event without a window-size limitation. The pipeline creates a future state and only reruns dependent instructions accumulated since the original checkpoint. The pipeline is capable of adaptively using checkpoints. While one thread is completing the future state created by the ahead thread, it continues execution to create the next future version of the architectural state. When the first future is completed, the thread proceeds to complete the next future that was being created. This leapfrogging continues until program execution is finished [3].

Cycle time is a factor that contributes to single-thread performance but also affects overall power dissipation. In this design, the cycle time is set by the pipeline, which is architected for 10 to 12 levels of logic gates per stage. The selected cycle time yields a good balance between pipeline depth, power and performance.

The physical address of the D$ is predicted from a 2-way skewed-associative cache of translations. This saves a full clock cycle by removing a 64b comparator from the critical path. This LRU-skewed cache has conflict detection on the predictor and uses the skewed aspect to remove conflicts. The L2$ provides an early signal so that restart of the pipeline is hidden from the effective latency of a load. The shared L2$ is banked to support high throughput. The 4 banks perform stores out-of-order to keep stores pipelined.

Load values in the future architectural state also appear in memory order when the future state is upgraded to the architectural state, i.e., they appear to be executed atomically. These features and the fact that the architectural state is moved atomically or discarded fully, make support for transactional memory [4] a natural fit.

Transactional memory is the ability to perform a set of instructions atomically (a transaction). New instructions and microarchitectural structures enable the execution of transactions without expensive atomic instructions. This enables multiple threads to enter a critical section simultaneously and allows programmers to write applications based on transactions as opposed to complex locks.

All clean structures like the I$ and D$ are parity protected while dirty structures like the L2$ and architecture register files are ECC protected. The FPU datapaths are protected by modulo arithmetic. External interfaces support pin sparing and a CRC-based retry protocol.

Linear scalability across multiple chips is achieved through (a) a directory-based cache coherency protocol, (b) point-to-point communication between all processors and all L3$/memory controllers, (c) uniform memory accesses—all DIMMs are at the same distance, (d) extensive pre-fetching of locks, critical section code and data through scout threading and (e) hardware support for transactional memory. The number of links needed to connect to multiple sockets and the bandwidth requirements result in 3 sides of the die being almost exclusively devoted to 256 SerDes operating at 2.67Gb/s (Fig. 4.1.3) and delivering over 80GB/s of bandwidth.

References:

[1] G. Konstadinidis, M. Rashid, P. Lai et al., "Third-Generation 16-core 32-Thread CMT SPARC® Processor", *ISSCC Dig. Tech. Papers*, pp. AA-BB , Feb. 2008.

[2] S. Chaudhry, P. Caprioli, S. Yip, M. Tremblay, "High-performance Throughput Computing," *IEEE Micro*, vol. 25, no. 3, pp. 32-45, May 2005.

[3] Marc Tremblay, "A Modern High-Performance Processor Pipeline", Keynote at *International Conference on Supercomputing* (ICS), Cairns, Australia, June 2006

[4] Marc Tremblay, "Transactional Memory for a Modern Microprocessor", Keynote at *Principles of Distributed Computing (PODC)*, Portland, USA, August 2007

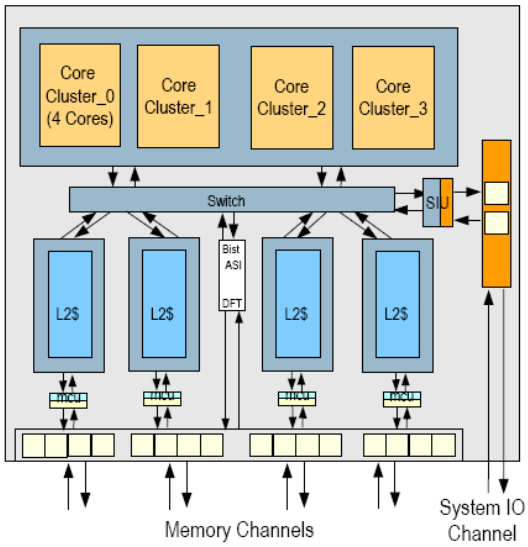

Figure 4.1.1: Top level logical block diagram.

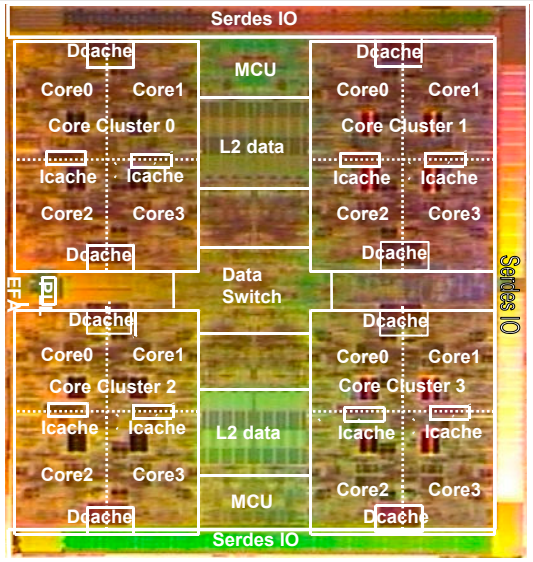

Figure 4.1.3: Chip micrograph.

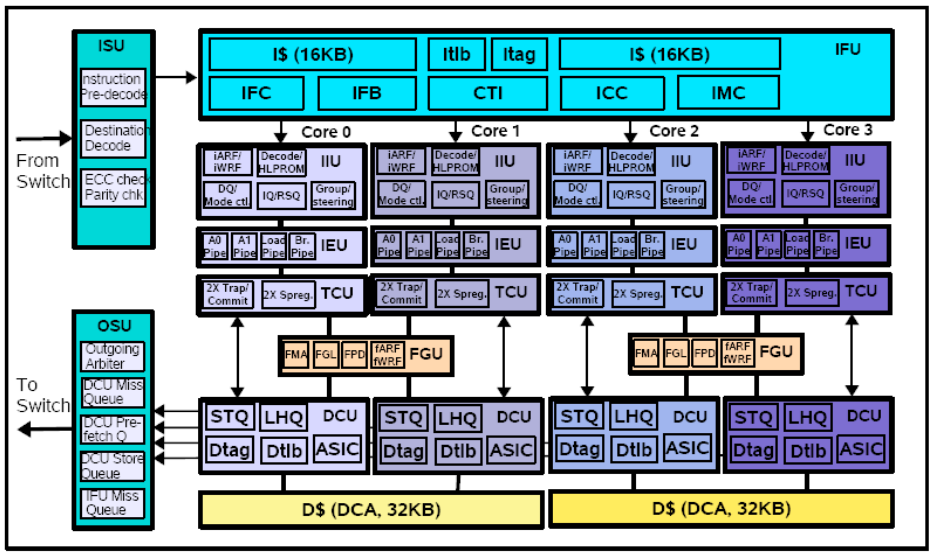

Figure 4.1.2: Core Cluster (4 cores).

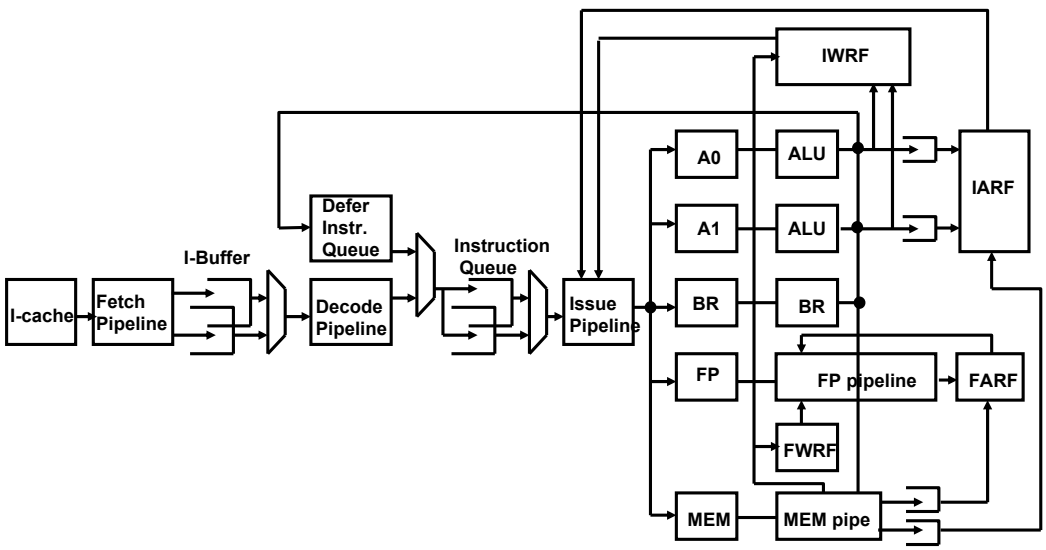

Figure 4.1.4: Processor Pipeline.